Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Стандартные обобщенные алгоритмы библиотеки STL»

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Желнин Н.А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

Постановка задачи

Вариант 14

Задача 1:

1. Создать последовательный контейнер.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции;

3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill());

4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy());

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort());

6. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if());

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) ;

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Указания варианта:

1. Контейнер - вектор; 2. Тип элементов - Pair

Задача 2:

1. Создать адаптер контейнера;

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции;

3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill());

4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy());

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort());

6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if());

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) ;

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Указание варианта:

Адаптер контейнера – очередь.

Задача 3

1. Создать ассоциативный контейнер;

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции;

3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill());

4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy());

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort());

6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if());

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each());

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Указание варианта:

Ассоциативный контейнер – множество с дубликатами.

Дополнение варианта ко всем задачам:

1. Найти минимальный элемент и добавить его на заданную позицию контейнера;

2. Найти меньше среднего арифметического и удалить их из контейнера;

3. Каждый элемент разделить на максимальный элемент контейнера.

Код программы

Задание 1

#include <iostream>

#include <vector>

#include <algorithm>

#include <functional>

#include "pair.h"

using namespace std;

struct Comp\_less{

public:

bool operator() (Pair p1, Pair p2){

if (p1>p2) return true;

else return false;

}

};

ostream& operator<<(ostream& out, vector<Pair>& v) {

for\_each(v.begin(), v.end(), [](Pair& p) {cout << p; });

return out;

}

bool EQUAL(Pair a, Pair t) {

if (a == t) return true;

return false;

}

void menu(const int c, vector<Pair>& v) {

switch (c) {

//sort

case 1: {

cout << v << endl;

cout << "< - po vozr\n> - po ubiv\n";

char op;

cin >> op;

if (op == '<') sort(v.begin(), v.end());

else if (op == '>') sort(v.begin(), v.end(), Comp\_less());

else cout << "wrong character\n";

cout << v << endl;

break;

}

//find element

case 2: {

cout << v << "\n\n";

cout << "Input value:\n";

Pair value;

cin >> value;

auto it = find\_if(v.begin(), v.end(), bind2nd(ptr\_fun(EQUAL), value));

if (it == v.end()) {

cout << "no such element!" << endl;

break;

}

for (auto i = v.begin(); i != v.end(); ++i) {

if (i == it) {

cout << "the element is " << \*i << endl;

}

}

break;

}

//delete < sred

case 3: {

cout << v << endl << endl;

Pair sum;

for\_each(v.begin(), v.end(), [&sum](Pair& p) {sum += p; });

sum.set\_first(sum.get\_first() / v.size());

sum.set\_second(sum.get\_second() / v.size());

for (vector<Pair>::iterator i = v.begin(); i != v.end(); ++i) {

i = v.erase(i);

}

break;

cout << v << endl;

break;

}

//dobavit min

case 4: {

cout << v << endl;

auto mini = min\_element(v.begin(), v.end());

//cout << "Summary of each elment: " << endl;

//for\_each(v.begin(), v.end(), [](Pair& p) {cout << p.sum() << ' '; });

int pos;

cout << "\nPosition: "; cin >> pos;

auto iter = v.cbegin();

v.emplace(iter+pos-1, mini);

//auto s = remove\_if(v.begin(), v.end(), [one, two](Pair& p) {return p >= one && p <= two; });

//v.erase(s, v.end());

cout << "Elements: " << endl;

for\_each(v.begin(), v.end(), [](Pair& p) {cout << p; });

break;

}

//delenie na max

case 5: {

cout << v << endl << endl;

//auto mini = min\_element(v.begin(), v.end());

auto maxi = max\_element(v.begin(), v.end());

//cout << "min: " << \*mini;

cout << "max: " << \*maxi;

int tmp;

//sum = \*mini + \*maxi;

//cout << "summary = " << sum << endl;

for\_each(v.begin(), v.end(), [&maxi](Pair& p) {p /= \*maxi;});

cout << endl << v << endl;

break;

}

//print

case 6: {

cout << v << endl;

break;

}

//exit

default: {

cout << "\nExiting from menu" << endl;

exit(777);

}

}

}

int main() {

int n;

cout << "Size: ";

cin >> n;

vector<Pair> v(n);

generate(v.begin(), v.end(), []() {Pair p; p.set\_random(); return p; });

cout << v << endl;

while (true) {

system("pause");

system("cls");

cout << "1. Sort container\n";

cout << "2. Find a pair\n";

cout << "3. Delete < sred\n";

cout << "4. Dobavit min\n";

cout << "5. Delenie na max\n";

cout << "6. Print container\n";

int c;

cin >> c;

system("cls");

menu(c, v);

}

return 0;

}

#pragma once

#include <iostream>

#include <string>

#include <cmath>

#include <fstream>

using namespace std;

class Pair {

private:

int first;

double second;

public:

Pair() { first = 0; second = 0; };

Pair(int F, double S) { first = F; second = S; };

Pair(const Pair& t) { first = t.first;second = t.second; };

~Pair() {};

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int F) { first = F;}

void set\_second(double S) { second = S;}

void set\_random();

double sum() { double s = second + first; return s; }

Pair operator =(const Pair&);

bool operator <(const Pair&) const;

bool operator >(const Pair&) const;

bool operator <=(const Pair&) const;

bool operator >=(const Pair&) const;

bool operator ==(const Pair&) const;

Pair& operator -(const Pair& p);

Pair& operator +(const Pair& p);

Pair& operator +=(const Pair& p);

Pair& operator +(const int X);

Pair& operator +(const double X);

Pair& operator /(const int X);

Pair& operator /=(const Pair& p);

friend istream& operator>>(istream& in, Pair& p);

friend ostream& operator<<(ostream& out, const Pair& p);

friend fstream& operator>>(fstream& fin, Pair& p);

friend fstream& operator<<(fstream& fout, const Pair& p);

};

Pair& Pair::operator +(const int X) {

first += X;

return \*this;

}

Pair& Pair::operator +(const double X) {

second += X;

return \*this;

}

Pair& Pair::operator /(const int X) {

first = first / X;

second = second / X;

return \*this;

}

Pair& Pair::operator /=(const Pair& p) {

this->first /= p.first;

this->second /= p.second;

return \*this;

}

Pair& Pair::operator -(const Pair& p) {

this->first -= p.first;

this->second -= p.second;

return \*this;

}

Pair& Pair::operator +(const Pair& p) {

this->first += p.first;

this->second += p.second;

return \*this;

}

Pair& Pair::operator +=(const Pair& p) {

this->first += p.first;

this->second += p.second;

return \*this;

}

istream& operator >>(istream& in, Pair& t) {

cout << "int first = "; in >> t.first;

cout << "double second = "; in >> t.second;

return in;

}

ostream& operator <<(ostream& out, const Pair& t) {

return (out << t.first << " : " << t.second << endl);

}

Pair Pair::operator =(const Pair& p) {

if (&p == this) return \*this;

first = p.first;

second = p.second;

return\*this;

}

fstream& operator>>(fstream& fin, Pair& p) {

fin >> p.first;

fin >> p.second;

return fin;

}

fstream& operator<<(fstream& fout, const Pair& p) {

fout << p.first << "\n" << p.second << "\n";

return fout;

}

bool Pair::operator <(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis < sother) return true;

return false;

}

bool Pair::operator >(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis > sother) return true;

return false;

}

bool Pair::operator <=(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis <= sother) return true;

return false;

}

bool Pair::operator >=(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis >= sother) return true;

return false;

}

bool Pair::operator ==(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis == sother) return true;

return false;

}

void Pair::set\_random() {

first = rand() % 10 + 1;

second = round(((double)(rand()) / RAND\_MAX \* 10 + 1) \* 100) / 100;

}

Задание 2

//MAIN

#include <iostream>

#include <queue>

#include <vector>

#include <algorithm>

#include <functional>

#include "pair.h"

using namespace std;

typedef queue<Pair> Q;

typedef vector<Pair> V;

struct Comp\_less{

public:

bool operator() (Pair p1, Pair p2){

if (p1>p2) return true;

else return false;

}

};

Q make\_q(int n) {

Q s;

Pair t;

for (int i = 0;i < n;i++) {

cin >> t;

s.push(t);

}

return s;

}

V copy\_q\_to\_v(Q q) {

V v;

while (!q.empty()) {

v.push\_back(q.front());

q.pop();

}

return v;

}

Q copy\_v\_to\_q(V v) {

Q q;

for (auto it = v.cbegin();it != v.cend();++it)

q.push(\*it);

return q;

}

ostream& operator<<(ostream& out, Q q) {

V v = copy\_q\_to\_v(q);

for\_each(v.begin(), v.end(), [](Pair& p) {cout << p; });

return out;

}

bool EQUAL(Pair a, Pair t) {

if (a == t) return true;

return false;

}

void menu(const int c, Q q) {

switch (c) {

//sort

case 1: {

cout << q << endl;

cout << "< - po vozr\n> - po ubiv\n";

char op;

cin >> op;

V v = copy\_q\_to\_v(q);

if (op == '<') sort(v.begin(), v.end());

else if (op == '>') sort(v.begin(), v.end(), Comp\_less());

else cout << "wrong character\n";

cout << q << endl;

break;

}

//find element

case 2: {

cout << q << "\n\n";

cout << "Input value:\n";

Pair value;

cin >> value;

V v = copy\_q\_to\_v(q);

auto it = find\_if(v.begin(), v.end(), bind2nd(ptr\_fun(EQUAL), value));

if (it == v.end()) {

cout << "no such element!" << endl;

break;

}

for (auto i = v.begin(); i != v.end(); ++i) {

if (i == it) {

cout << "the element is " << \*i << endl;

}

}

break;

}

//delete < sred

case 3: {

cout << q << endl << endl;

Pair sum;

V v = copy\_q\_to\_v(q);

for\_each(v.begin(), v.end(), [&sum](Pair& p) {sum += p; });

sum.set\_first(sum.get\_first() / v.size());

sum.set\_second(sum.get\_second() / v.size());

for (vector<Pair>::iterator i = v.begin(); i != v.end(); ++i) {

i = v.erase(i);

}

break;

cout << q << endl;

break;

}

//dobavit min

case 4: {

cout << q << endl;

V v = copy\_q\_to\_v(q);

auto mini = min\_element(v.begin(), v.end());

//cout << "Summary of each elment: " << endl;

//for\_each(v.begin(), v.end(), [](Pair& p) {cout << p.sum() << ' '; });

int pos;

cout << "\nPosition: "; cin >> pos;

auto iter = v.cbegin();

v.emplace(iter+pos-1, mini);

//auto s = remove\_if(v.begin(), v.end(), [one, two](Pair& p) {return p >= one && p <= two; });

//v.erase(s, v.end());

cout << "Elements: " << endl;

for\_each(v.begin(), v.end(), [](Pair& p) {cout << p; });

break;

}

//delenie na max

case 5: {

cout << q << endl << endl;

V v = copy\_q\_to\_v(q);

//auto mini = min\_element(v.begin(), v.end());

auto maxi = max\_element(v.begin(), v.end());

//cout << "min: " << \*mini;

cout << "max: " << \*maxi;

int tmp;

//sum = \*mini + \*maxi;

//cout << "summary = " << sum << endl;

for\_each(v.begin(), v.end(), [&maxi](Pair& p) {p /= \*maxi;});

cout << endl << q << endl;

break;

}

//print

case 6: {

cout << q << endl;

break;

}

//exit

default: {

cout << "\nExiting from menu" << endl;

exit(777);

}

}

}

int main() {

int n;

cout << "Size: ";

cin >> n;

Q q;

q = make\_q(n);

V v = copy\_q\_to\_v(q);

generate(v.begin(), v.end(), []() {Pair p; p.set\_random(); return p; });

cout << q << endl;

while (true) {

system("pause");

system("cls");

cout << "1. Sort container\n";

cout << "2. Find a pair\n";

cout << "3. Delete < sred\n";

cout << "4. Dobavit min\n";

cout << "5. Delenie na max\n";

cout << "6. Print container\n";

int c;

cin >> c;

system("cls");

menu(c, q);

}

return 0;

}

//PAIR

#pragma once

#include <iostream>

#include <string>

#include <cmath>

#include <fstream>

using namespace std;

class Pair {

private:

int first;

double second;

public:

Pair() { first = 0; second = 0; };

Pair(int F, double S) { first = F; second = S; };

Pair(const Pair& t) { first = t.first;second = t.second; };

~Pair() {};

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int F) { first = F;}

void set\_second(double S) { second = S;}

void set\_random();

double sum() { double s = second + first; return s; }

Pair operator =(const Pair&);

bool operator <(const Pair&) const;

bool operator >(const Pair&) const;

bool operator <=(const Pair&) const;

bool operator >=(const Pair&) const;

bool operator ==(const Pair&) const;

Pair& operator -(const Pair& p);

Pair& operator +(const Pair& p);

Pair& operator +=(const Pair& p);

Pair& operator +(const int X);

Pair& operator +(const double X);

Pair& operator /(const int X);

Pair& operator /=(const Pair& p);

friend istream& operator>>(istream& in, Pair& p);

friend ostream& operator<<(ostream& out, const Pair& p);

friend fstream& operator>>(fstream& fin, Pair& p);

friend fstream& operator<<(fstream& fout, const Pair& p);

};

Pair& Pair::operator +(const int X) {

first += X;

return \*this;

}

Pair& Pair::operator +(const double X) {

second += X;

return \*this;

}

Pair& Pair::operator /(const int X) {

first = first / X;

second = second / X;

return \*this;

}

Pair& Pair::operator /=(const Pair& p) {

this->first /= p.first;

this->second /= p.second;

return \*this;

}

Pair& Pair::operator -(const Pair& p) {

this->first -= p.first;

this->second -= p.second;

return \*this;

}

Pair& Pair::operator +(const Pair& p) {

this->first += p.first;

this->second += p.second;

return \*this;

}

Pair& Pair::operator +=(const Pair& p) {

this->first += p.first;

this->second += p.second;

return \*this;

}

istream& operator >>(istream& in, Pair& t) {

cout << "int first = "; in >> t.first;

cout << "double second = "; in >> t.second;

return in;

}

ostream& operator <<(ostream& out, const Pair& t) {

return (out << t.first << " : " << t.second << endl);

}

Pair Pair::operator =(const Pair& p) {

if (&p == this) return \*this;

first = p.first;

second = p.second;

return\*this;

}

fstream& operator>>(fstream& fin, Pair& p) {

fin >> p.first;

fin >> p.second;

return fin;

}

fstream& operator<<(fstream& fout, const Pair& p) {

fout << p.first << "\n" << p.second << "\n";

return fout;

}

bool Pair::operator <(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis < sother) return true;

return false;

}

bool Pair::operator >(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis > sother) return true;

return false;

}

bool Pair::operator <=(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis <= sother) return true;

return false;

}

bool Pair::operator >=(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis >= sother) return true;

return false;

}

bool Pair::operator ==(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis == sother) return true;

return false;

}

void Pair::set\_random() {

first = rand() % 10 + 1;

second = round(((double)(rand()) / RAND\_MAX \* 10 + 1) \* 100) / 100;

}

Задание 3

#include <iostream>

#include <set>

#include <vector>

#include <algorithm>

#include <functional>

#include "pair.h"

using namespace std;

typedef multiset<Pair> Q;

typedef vector<Pair> V;

struct Comp\_less{

public:

bool operator() (Pair p1, Pair p2){

if (p1>p2) return true;

else return false;

}

};

/\*

ostream& operator<<(ostream& out, Q v) {

//V v = copy\_q\_to\_v(q);

for\_each(v.begin(), v.end(), [](Pair& p) {cout << p; });

return out;

}

\*/

void Print(Q v){

multiset<Pair>::iterator i;

for(i = v.begin(); i!=v.end(); ++i) {

cout << i->first << " : " << i->second << endl;

}

}

bool EQUAL(Pair a, Pair t) {

if (a == t) return true;

return false;

}

void menu(const int c, Q v) {

switch (c) {

//sort

case 1: {

Print(v);

cout << "< - po vozr\n> - po ubiv\n";

char op;

cin >> op;

//V v = copy\_q\_to\_v(q);

if (op == '<') sort(v.begin(), v.end());

else if (op == '>') sort(v.begin(), v.end(), Comp\_less());

else cout << "wrong character\n";

Print(v);

break;

}

//find element

case 2: {

Print(v);

cout << "Input value:\n";

Pair value;

cin >> value;

//V v = copy\_q\_to\_v(q);

auto it = find\_if(v.begin(), v.end(), bind2nd(ptr\_fun(EQUAL), value));

if (it == v.end()) {

cout << "no such element!" << endl;

break;

}

for (auto i = v.begin(); i != v.end(); ++i) {

if (i == it) {

cout << "the element is " << \*i << endl;

}

}

break;

}

//delete < sred

case 3: {

Print(v);

Pair sum;

//V v = copy\_q\_to\_v(q);

for\_each(v.begin(), v.end(), [&sum](Pair& p) {sum += p; });

sum.set\_first(sum.get\_first() / v.size());

sum.set\_second(sum.get\_second() / v.size());

for (multiset<Pair>::iterator i = v.begin(); i != v.end(); ++i) {

i = v.erase(i);

}

break;

Print(v);

break;

}

//dobavit min

case 4: {

Print(v);

//V v = copy\_q\_to\_v(q);

auto mini = min\_element(v.begin(), v.end());

//cout << "Summary of each elment: " << endl;

//for\_each(v.begin(), v.end(), [](Pair& p) {cout << p.sum() << ' '; });

int pos;

cout << "\nPosition: "; cin >> pos;

//auto iter = v.cbegin();

v.insert(\*mini);

//auto s = remove\_if(v.begin(), v.end(), [one, two](Pair& p) {return p >= one && p <= two; });

//v.erase(s, v.end());

cout << "Elements: " << endl;

for\_each(v.begin(), v.end(), [](Pair& p) {cout << p; });

break;

}

//delenie na max

case 5: {

Print(v);

//V v = copy\_q\_to\_v(q);

//auto mini = min\_element(v.begin(), v.end());

auto maxi = max\_element(v.begin(), v.end());

//cout << "min: " << \*mini;

cout << "max: " << \*maxi;

int tmp;

//sum = \*mini + \*maxi;

//cout << "summary = " << sum << endl;

for\_each(v.begin(), v.end(), [&maxi](Pair& p) {p /= \*maxi;});

Print(v);

break;

}

//print

case 6: {

Print(v);

break;

}

//exit

default: {

cout << "\nExiting from menu" << endl;

exit(777);

}

}

}

int main() {

int n;

cout << "Size: ";

cin >> n;

Q q;

Pair a;

for(int i = 0; i<n; i++){

cin >> a;

q.insert(a);

}

//q = make\_q(n);

//V v = copy\_q\_to\_v(q);

//generate(v.begin(), v.end(), []() {Pair p; p.set\_random(); return p; });

Print(q);

while (true) {

system("pause");

system("cls");

cout << "1. Sort container\n";

cout << "2. Find a pair\n";

cout << "3. Delete < sred\n";

cout << "4. Dobavit min\n";

cout << "5. Delenie na max\n";

cout << "6. Print container\n";

int c;

cin >> c;

system("cls");

menu(c, q);

}

return 0;

}

#pragma once

#include <iostream>

#include <string>

#include <cmath>

#include <fstream>

using namespace std;

class Pair {

public:

int first;

double second;

Pair() { first = 0; second = 0; };

Pair(int F, double S) { first = F; second = S; };

Pair(const Pair& t) { first = t.first;second = t.second; };

~Pair() {};

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int F) { first = F;}

void set\_second(double S) { second = S;}

void set\_random();

double sum() { double s = second + first; return s; }

Pair operator =(const Pair&);

bool operator <(const Pair&) const;

bool operator >(const Pair&) const;

bool operator <=(const Pair&) const;

bool operator >=(const Pair&) const;

bool operator ==(const Pair&) const;

Pair& operator -(const Pair& p);

Pair& operator +(const Pair& p);

Pair& operator +=(const Pair& p);

Pair& operator +(const int X);

Pair& operator +(const double X);

Pair& operator /(const int X);

Pair& operator /=(const Pair& p);

friend istream& operator>>(istream& in, Pair& p);

friend ostream& operator<<(ostream& out, const Pair& p);

friend fstream& operator>>(fstream& fin, Pair& p);

friend fstream& operator<<(fstream& fout, const Pair& p);

};

Pair& Pair::operator +(const int X) {

first += X;

return \*this;

}

Pair& Pair::operator +(const double X) {

second += X;

return \*this;

}

Pair& Pair::operator /(const int X) {

first = first / X;

second = second / X;

return \*this;

}

Pair& Pair::operator /=(const Pair& p) {

this->first /= p.first;

this->second /= p.second;

return \*this;

}

Pair& Pair::operator -(const Pair& p) {

this->first -= p.first;

this->second -= p.second;

return \*this;

}

Pair& Pair::operator +(const Pair& p) {

this->first += p.first;

this->second += p.second;

return \*this;

}

Pair& Pair::operator +=(const Pair& p) {

this->first += p.first;

this->second += p.second;

return \*this;

}

istream& operator >>(istream& in, Pair& t) {

cout << "int first = "; in >> t.first;

cout << "double second = "; in >> t.second;

return in;

}

ostream& operator <<(ostream& out, const Pair& t) {

return (out << t.first << " : " << t.second << endl);

}

Pair Pair::operator =(const Pair& p) {

if (&p == this) return \*this;

first = p.first;

second = p.second;

return\*this;

}

fstream& operator>>(fstream& fin, Pair& p) {

fin >> p.first;

fin >> p.second;

return fin;

}

fstream& operator<<(fstream& fout, const Pair& p) {

fout << p.first << "\n" << p.second << "\n";

return fout;

}

bool Pair::operator <(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis < sother) return true;

return false;

}

bool Pair::operator >(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis > sother) return true;

return false;

}

bool Pair::operator <=(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis <= sother) return true;

return false;

}

bool Pair::operator >=(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis >= sother) return true;

return false;

}

bool Pair::operator ==(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis == sother) return true;

return false;

}

void Pair::set\_random() {

first = rand() % 10 + 1;

second = round(((double)(rand()) / RAND\_MAX \* 10 + 1) \* 100) / 100;

}

Диаграмма и вывод в консоль

